

## Aufgabe Agar.io

Arbeitszeit 60 min

### **Einleitung:**

Implementiere ein Teil einer Einspieler-Version von agar.io. Folge dabei den unten stehenden Anweisungen! Du darfst für deine Lösung jederzeit deine Unterlagen, den Kurs im Mebis und jedes Online-Mittel verwenden. Sorge aber dafür, dass die Lösung den Anweisungen entspricht! Speichere regelmäßig und häufig deine Arbeit!

### **Aufgabe 1:**

**4 BE**

Öffne Processing und speichere einen neuen Sketch mit dem Namen `sketch_agario_deineNachname`. In diesem Sketch bereite den Hauptreiter (Haupttab) mit einem `setup`- und einem `draw`-Block vor. Setze die Größe der Leinwand auf 800 mal 600 Pixel. Lass Processing am Anfang des `draw`-Blocks den Hintergrund in einer Farbe deiner Wahl zeichnen.

### **Aufgabe 2:**

**3+4+3 BE**

2a) Definiere **in einem neuen Reiter namens Spieler** eine Klasse `Spieler` mit einem Attribut `radius` vom Datentyp `float`. Definiere für diese Klasse einen Konstruktor, der diesem Attribut den Anfangswert 50 zuweist.

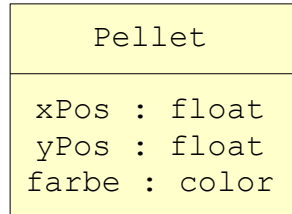
2b) Definiere in der Klasse `Spieler` eine Methode `anzeigen()`, die den Spieler als Kreis auf der Leinwand zeichnet. Der Kreis soll immer genau in der Mitte der Leinwand gezeichnet werden. Der Durchmesser des Kreises soll doppelt so viel wie der Wert im Attribut `radius` sein (bitte Attribut dafür verwenden und nicht direkt den Wert 50!). Die Farbe darfst du selbst auswählen.

2c) Definiere ganz am Anfang des Hauptreiters eine Variable `s` vom Datentyp `Spieler`. Erzeuge im `setup`-Block ein Objekt der Klasse `Spieler` und weise dieses Objekt der Variablen `s` zu. Rufe im `draw`-Block (mit der Punktnotation) die Methode `anzeigen()` des Objekts `s` auf.

Teste, ob jetzt einen Kreis der richtigen Größe in der Mitte der Leinwand gezeichnet wird!

**Aufgabe 3:****2+3+4+2+3+3 BE**

3a) Definiere in einem neuen Reiter namens Pellet eine Klasse `Pellet` nach dem folgenden Klassendiagramm:



3b) Definiere in der Klasse `Pellet` einen Konstruktor mit **Eingabeparametern** für die Attribute `xPos` und `yPos`. Weise im Konstruktor dem Attribut `farbe` eine zufällige Farbe zu.

3c) Definiere in der Klasse `Pellet` eine Methode `anzeigen()`, die ein `Pellet`-Objekt auf der Leinwand als Kreis mit der im Attribut `farbe` gespeicherten Farbe zeichnet. Der Durchmesser des Kreises ist 20.

3d) Definiere im Hauptreiter eine Variable `pellets` von **Datentyp Array von Pellet(-Objekte)**. Weise im `setup`-Block dieser Variable ein neues (leeres) Array von Objekten der Klasse `Pellet` zu. Das Array soll die Länge 60 haben.

3e) Mit einer `for`-Schleife im `setup`-Block weise jeder Position des Array `pellets` ein neu erzeugtes `Pellet`-Objekt zu. Beim Erzeugen eines neuen `Pellet`-Objektes verwende `random(width)` und `random(height)` um zufällige Werte für die Konstruktor-Eingaben zu bestimmen.

3f) Rufe im `draw`-Block mit einer `for`-Schleife die Methode `anzeigen()` aller im `pellets` gespeicherte `Pellet`-Objekte auf.

Jetzt solltest du beim Abspielen deines Sketches das Spieler-Objekt samt 60 `Pellet`-Objekten auf der Leinwand sehen. Du hast schon einen guten Teil des Test abgeschlossen!

**Aufgabe 4:****3+3+2+3+2 BE**

4a) Ergänze die Klasse `Spieler` mit einem Attribut `geschwindigkeit` vom Datentyp `float` und einem Attribut `richtung` vom Datentyp **PVector** (s. *Processing Language Reference*). Weise im Konstruktor dem Attribut `geschwindigkeit` den Anfangswert 1 und dem Attribut `richtung` den Anfangswert `new PVector(mouseX, mouseY)`. Ändere (immer noch im Konstruktor unmittelbar danach) den Betrag des Vektors `richtung` mit dem Befehl `richtung.setMag(geschwindigkeit)`.

4b) Definiere in der Klasse `Spieler` eine Methode `aktualisieren()`, die mit den zwei Aufrufen `richtung.set(width/2-mouseX, height/2-mouseY)` und `richtung.setMag(geschwindigkeit)` den Wert des Attributs `richtung` mit der neuen Position der Maus aktualisiert. Rufe diese Methode im `draw-Block` auf.

4c) Definiere in der Klasse `Spieler` eine **Methode mit Rückgabewert** `gibRichtung()`, die den `PVector`-Objekt im Attribut `richtung` zurückgibt.

4d) Wir möchten den Spieler mit der Maus steuern. Das Ergebnis soll aber sein, dass der Spieler immer in der Mitte der Leinwand bleibt, während sich die Umgebung anpasst, um den Eindruck zu erwecken, der Spieler-Kreis folge der Maus. Um das zu erreichen, müssen wir die Position aller Pellet-Objekte um den Wert, der dem `Pvector` `richtung` des Spielers entspricht, ändern. Definiere dazu in der Klasse `Pellet` eine Methode mit Eingabeparameter `aktualisieren(PVector v)`, die die Werte der Attribute `xPos` und `yPos` um `v.x` beziehungsweise `v.y` ändert.

4e) Rufe im `draw-Block` die Methode `aktualisieren` aller `Pellet`-Objekte **mit der Eingabe** `s.gibRichtung()` auf.

Weitere Entwicklung: Lass den Spieler die Pellets fressen. Weise einem gefressenen Pellet eine neue zufällige Position zu, so dass der Eindruck entsteht, die Pellet werden ab und zu neu generiert.

Gut gemacht!